**National Cheng Kung University**

**Department of Electrical Engineering**

***Introduction to VLSI CAD (Spring 2020)***

**Lab Session 7**

**Design of CNN Processing System**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student ID | | |
| 王嘉瑋 | E24079075 | | |
| Practical | | Points | Marks |
| Part 1 | | 10 |  |
| Part 2 | | 10 |  |
| Part 3 | | 15 |  |
| Part 4 | | 15 |  |
| Part 5  (Word整體報告內容詳細程度) | | 10 |  |
| Notes | |  |  |
|  | | | |

**Due: 23:50 May 3, 2020@ moodle**

**Deliverables**

1. All Verilog codes including testbenche should be uploaded.

NOTE: Please **DO NOT** include source code in the paper report!

1. All homework requirements should be uploaded in this file hierarchy.
2. Please **DO NOT** upload waveforms (.fsdb or .vcd)!
3. **If you upload a dead body which we can’t even compile, you will get NO credit!**
4. **All Verilog file should get at least 90% SuperLint Coverage.**
5. All homework requirements should be uploaded in this file hierarchy or you will not get full credit.

**File Hierarchy**

* Lab7\_E240XXXXX\_E240XXXXX.tar(Lab7\_E240XXXXX.tar) (Don’t add other text in file name)
  + Lab7\_E240XXXXX\_E240XXXXX (Main folder of this project)
    - Lab7\_E240XXXXX\_E240XXXXX.docx (Your homework report)
    - Makefile (You shouldn’t modify it)
* data (Images data in .txt format)
* test\_im\*.txt
* golden (Golden hexadecimal data)
* \*.txt
* images (Hand-written digit images)
* \*.png
* Include (RTL parameters definition file)
* def.v
* parameters (The weights and bias of the neural network)
  + layer\*\_w.txt
  + layer\*\_b.txt
* script (Any scripts of verification and synthesis)
  + - * Script files (DC.sdc, \*.tcl)
* sim
* top\_tb.v
* tsmc13\_neg.v
  + - * ROM (ROM behavior model)
        + ROM.v
        + ROM\_tb.v
      * TwoPortSRAM (Two-Port SRAM behavior model)
        + TwoPortSRAM.v
        + TwoPortSRAM\_tb.v
* src (RTL code)
* Adder.v
* Controller.v
* Decoder.v
* MUX2to1\_16b.v
* MUX2to1\_32b.v
* MUX4to1\_32b.v
* PE.v
* Pooling.v
* Relu.v
* Truncate.v
* top.v
* syn (Synthesized code and timing file)
* top\_syn.v
* top\_syn.sdf

*Part1*

* Objective:

Build a CNN Processing System as the architecture of Fig.1. The system can do inference of CNN as Fig.2 and classify hand-written digit images.

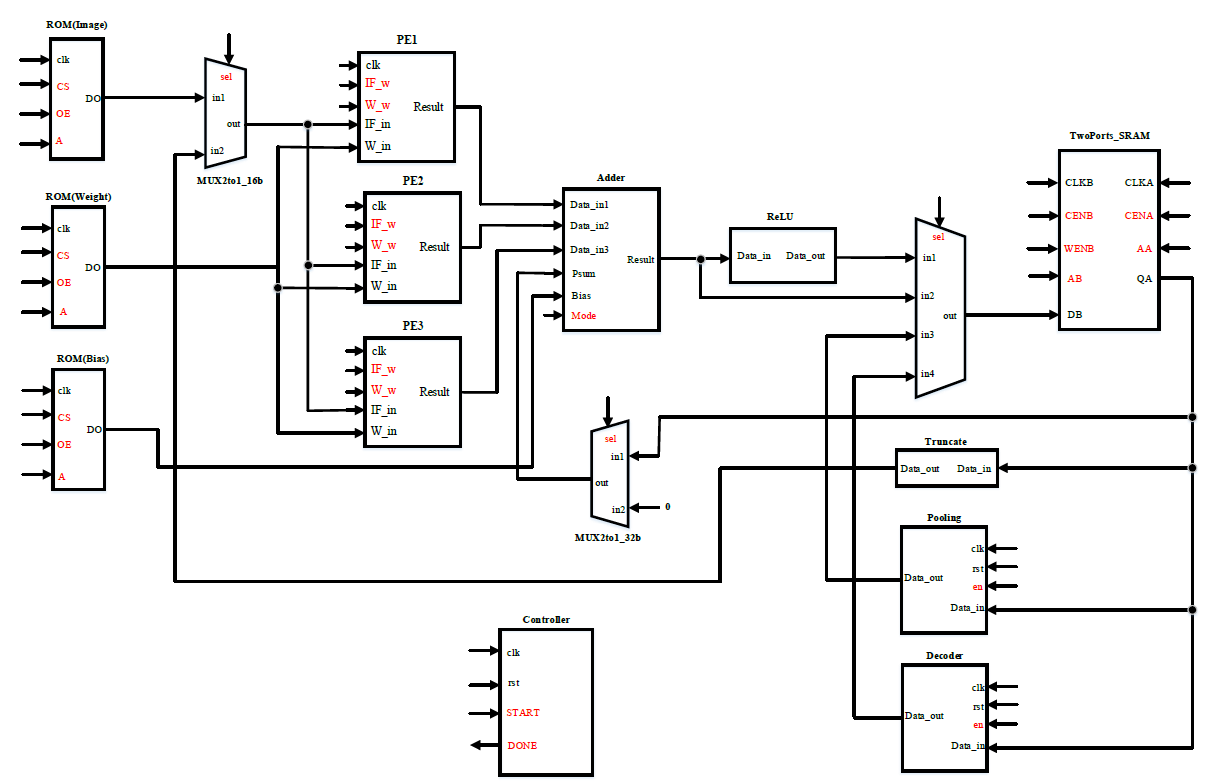


Fig.1 The architecture of CNN Processing System

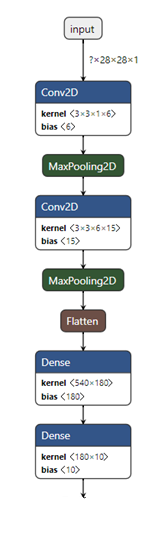


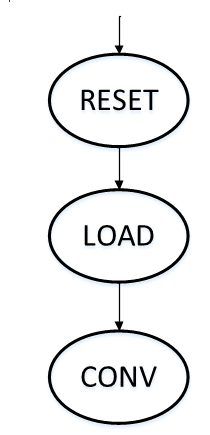
Fig.2 CNN NetworkArchitecture

* Please follow the tutorial of PowerPoint, and build a CNN Processing System by yourself. Run the system and let it pass the convolution layer 0 (Conv0) result for the image 0.

Simulation command: make rtl\_conv0

* Show your finite state machine (FSM) of the controller. Describe how your controller works.

Note: Please use software tools to clearly draw the FSM.

EX:

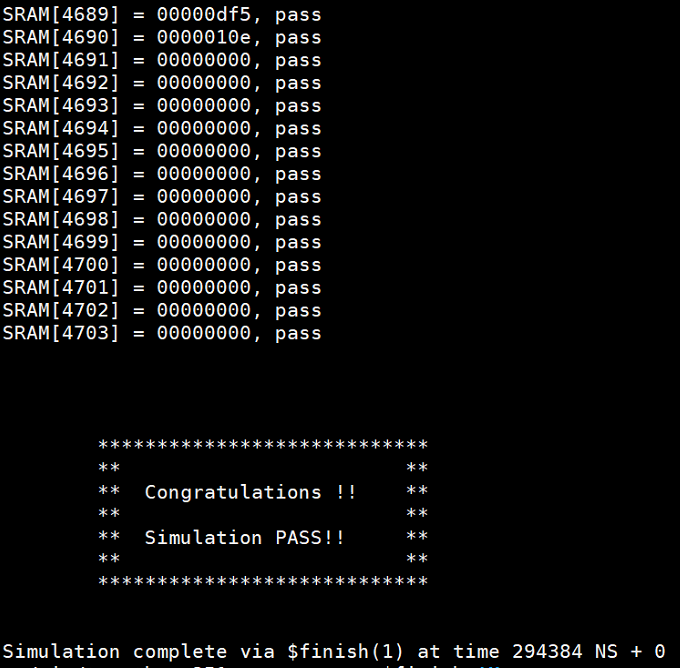
當Controller收到rst訊號時會進入RESET state，收到START訊號後會進入LOAD state。在LOAD state會去記憶體讀取convolution運算的資料(權重、partial sum、input feature map)，讀取完資料後進入CONV state做convolution運算。

|  |
| --- |
| Your FSM Design |
|  |

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

EX:



|  |
| --- |
| Simulation Result |
|  |

*Part2*

* Run the system and pass the pooling layer 1 (Pooling1) result of the image 0.

Simulation command: make rtl\_pool1

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

|  |
| --- |
| Simulation Result |
|  |

*Part3*

* Run the RTL code. Let the CNN Processing System classify 200 images.

Simulation command: make rtl\_full

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

|  |
| --- |
| Simulation Result |
|  |

*Part4*

* Run the post-synthesis simulation. Let the CNN Processing System classify 10 images.

Simulation command: make syn\_full

* Paste the simulation result on the terminal.

Note: Don’t paste all the results. You only need to paste the result of the simulation end.

|  |
| --- |
| Simulation Result |
|  |

*Part5*

* Show your Superlint coverage

Note: Use the following command to get the lines of your code.

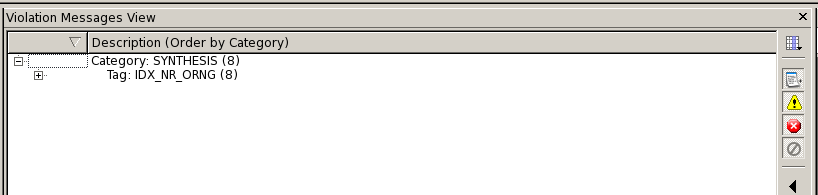
Command: wc –l ./src/\*

EX:

Number of lines of RTL codes:

Total lines = 1458

Number of warnings or errors on Superlint: 8



Coverage:

Coverage = 99.45%

|  |
| --- |
| Superlint Coverage |
| Coverage = 1 - 2/1799 = 99.9% |

* Lesson Learned

這次實驗真的被累到，而且想說自己試試看一個人打完，雖然做完了，但真的也花了不少時間，debug占了大部分的時間…，好處是學到了很多東西，而且有點成就感。很感謝助教這半學期的幫忙，常常幫我們作業想辦法，而且有問必答，真的是辛苦助教了!

* Contribution

EX: A 50 %, B 50 %

E24079075王嘉瑋 100%

*Appendix*

1. Simulation Requirements

You should make sure that your code can be simulated with specified commands in Table A-1. TA will use the same commands to check your design under SoC Lab environment. If your code can’t be recompiled by TA, you will get no credit.

TA will also see how many problems do you finish to decide which command TA will run. For example, if you only finish convolution layer 0, TA will run “make rtl\_conv0”. However, if you finish all layers, then TA will run “make rtl\_full”.

Table A-1: Simulation Commands

|  |  |
| --- | --- |
| **Command** | **Description** |
| make rtl\_conv0 | Run RTL simulation of convolution layer 0 for the image 0. |
| make rtl\_pool1 | Run RTL simulation of pooling layer 1 for the image 0. |
| make rtl\_full | Run RTL simulation of the CNN for 200 images. |
| make syn\_full | Run post-synthesis simulation of the CNN for 10 images. |